Partie 1 Séquence de démarrage

- Maitriser la séquence de démarrage des systèmes d'exploitation Linux.
- Configurer le gestionnaire de démarrage.

1-Séquence de démarrage :

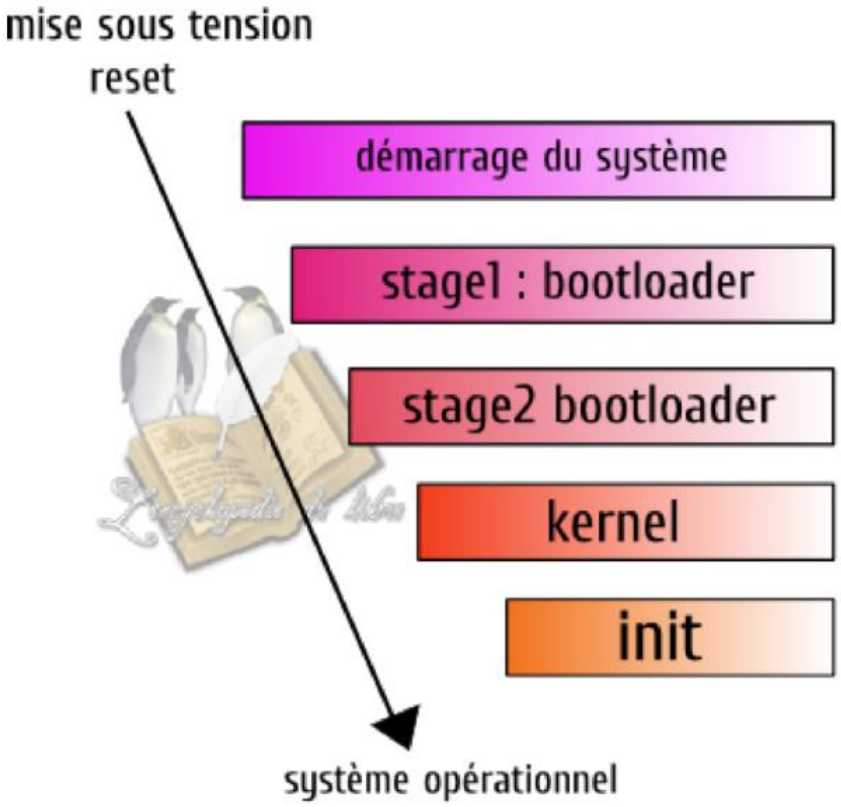
Définition : La séquence de démarrage c'est toutes les étapes qui vont être exécutées dès le moment où vous allez démarrer votre ordinateur.

Suite à la mise sous tension de l'ordinateur, la séquence de démarrage est déclenchée, qui comporte principalement les étapes suivantes :

● Localiser et lancer le chargeur de démarrage GRUB2

● Exécuter le noyau du système d'exploitation

● Lancer le premier processus



Un chargeur d'amorçage (bootloader en anglais) est un programme informatique qui se charge en mémoire au démarrage d'un ordinateur et qui permet de lancer le système d'exploitation installé sur le disque dur ou le SSD.

Le rôle principal du chargeur d'amorçage est de localiser les fichiers nécessaires au démarrage du système d'exploitation et de les charger en mémoire.

Le stage 1 bootloader est chargé par le microprogramme (BIOS ou UEFI) au moment du démarrage de l'ordinateur. Il est généralement stocké sur le secteur d'amorçage du disque dur et est conçu pour être suffisamment petit pour être chargé en quelques secteurs. Sa fonction principale est de localiser le stage 2 bootloader et de le charger en mémoire.

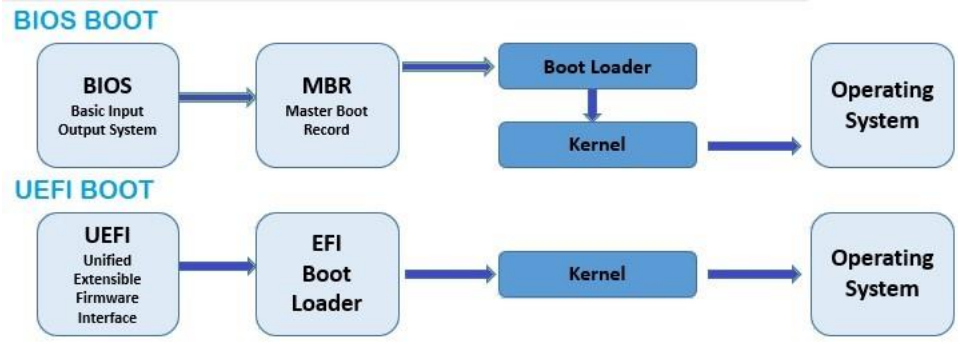
Le stage 2 bootloader est le chargeur d'amorçage principal qui contient le code nécessaire pour charger le système d'exploitation.

Le stage 2 bootloader peut être plus gros que le stage 1 bootloader, car il doit contenir des pilotes de périphériques pour les disques, les réseaux et d'autres matériels nécessaires pour démarrer le système d'exploitation.

Le noyau est responsable de la gestion des ressources matérielles, telles que les processeurs, la mémoire, les périphériques d'E/S (entrée/sortie) et les disques durs.

Init est responsable de plusieurs tâches clés lors du démarrage du système, telles que la configuration des variables d'environnement, le montage des systèmes de fichiers, l'initialisation des démons système.

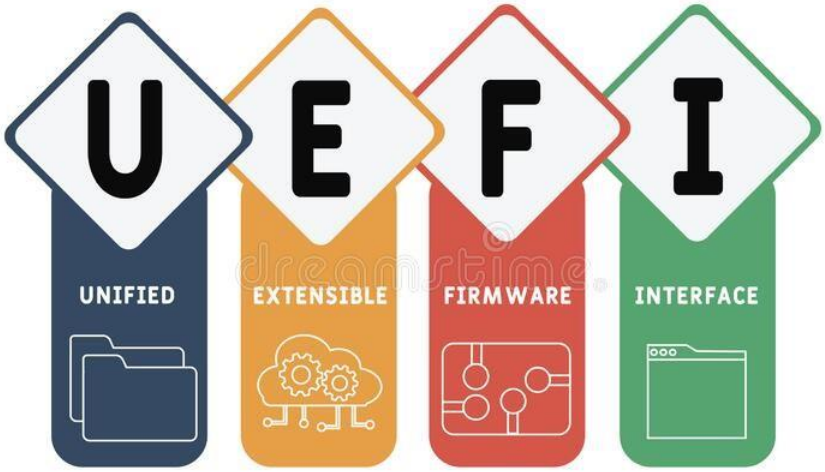
2- Limites MBR & BIOS :



Qu'est-ce que le BIOS	Qu'est-ce que l'UEFI
Le BIOS (Basic Input Output System) est un microprogramme avec de nombreuses fonctions, stockées dans la mémoire morte (ROM) de la carte mère d'un ordinateur, lui permettant d'effectuer des opérations élémentaires lors de sa mise sous tension et de charger le système d'exploitation.	UEFI remplace le BIOS depuis 2006. L'UEFI est comme le BIOS, un microprogramme qui vise à configurer votre carte mère, mais pas que. Il est stocké dans la mémoire flash de votre carte mère et comporte de nombreux avantages comparés à son ancêtre, le BIOS.

La table de partitionnement du MBR possède des limitations: ● 4 partitions primaires maximum. ● Taille d'une partition limitée à 2,2 To. ● Le Bios ne dispose que d'1 Mo d'espace mémoire pour s'exécuter. Il a ainsi du mal à initialiser plusieurs périphériques en même temps, ce qui ralentit le démarrage du PC	• UEFI prend en charge des tailles de disque jusqu'à **9 zettaoctets,** tandis que le BIOS ne prend en charge que 2,2 téraoctets. • UEFI fournit un temps de démarrage plus rapide. • UEFI s'exécute en mode 32 bits ou 64 bits, tandis que le BIOS s'exécute en mode 16 bits. L'UEFI est donc capable de fournir une interface graphique meilleure telque navigation avec la souris. • Prise en charge du Secure boot

3- UEFI: Unified Extensible Firmware Interface:



- l'UEFI est un microprogramme stocké dans une mémoire flash située sur la carte mère de l'ordinateur.
- UEFI est installé au moment de la fabrication et constitue le premier programme qui s'exécute lors du démarrage d'un ordinateur.
- UEFI détecte les périphériques candidats à démarrer (clavier/souris etc..)
- Le système de fichiers pris en charge par UEFI est basé sur le système de fichiers FAT(FAT12, FAT16 et FAT32).

Secure Boot est une fonctionnalité de sécurité qui a été introduite dans l'UEFI pour protéger les systèmes contre les logiciels malveillants et les attaques au niveau du bootloader.

4- NVRAM : Mémoire vive RAM non volatile :

La NVRAM est une mémoire vive RAM non volatile utilisée par UEFI pour stocker les variables qui doivent persister entre les démarrages.

La NVRAM contient une liste de variables dont les options de boot suivantes :

✓ Quel système est actuellement démarré ?

✓ L'ordre des entrées de boot

✓ Le timeout avant de booter sur l'entrée par défaut (suivant l'ordre précédent)

✓ La liste des entrées de démarrage

5- Efibootmgr (Gestionnaire de démarrage UEFI) :

EFIBootmgr est un utilitaire utilisable en ligne de commande permettant de gérer le chargeur de démarrage EFI, il permet de :

- Répertorier les entrées du gestionnaire de démarrage EFI
- Changer l'ordre des entrées
- Supprimer une entrée de démarrage
- Créer une entrée de démarrage
- Définir l'intervalle de temporisation du gestionnaire de démarrage EFI

```
root@souha-virtual-machine:~# efibootmgr
BootCurrent: 0004
Timeout: 2 seconds
BootOrder: 0004,0000,0001,0002,0003
Boot0000* EFI VMware Virtual SCSI Hard Drive (0.0)
Boot0001* EFI VMware Virtual SATA CDROM Drive (1.0)
Boot0002* EFI Network
Boot0003* EFI Internal Shell (Unsupported option)
Boot0004* ubuntu
```

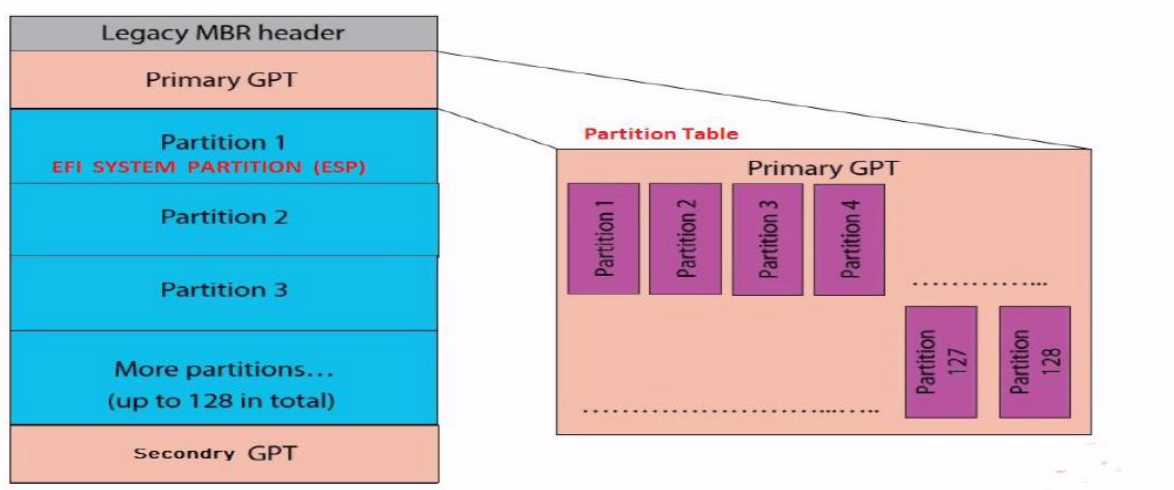
BootCurrent: indique l'entrée de démarrage utilisée pour démarrer le systeme. Dans notre cas, 0004 (Ubuntu).

Timeout: indique le temps avant l'amorçage. 2 secondes dans notre cas.

BootOrder: indique l'ordre dans lequel sont amorcés les systèmes UEFI.

efibootmgr –v	Pour afficher mes informations détaillées sur les entrées de démarrage EFI
efibootmgr –o	Pour modifier l'ordre de démarrage des entrées EFI

5- Structure GPT :



GPT (GUID Partition Table) est une structure de partitionnement de disque dur utilisée pour les disques de grande capacité et les disques SSD modernes.

- UEFI utilise un schéma de partitionnement en GPT « Globally Unique Identifier Partition Table » au lieu du MBR utilisé par le BIOS. Ce qui permettra d'utiliser des disques système de plus de 9 Zettaoctets.
- GPT soit Globally Unique Identifier Partition Table, est un standard pour décrire la table de partitionnement d'un disque dur.
- GPT utilise l'adressage logique des blocs **(LBA) UEFI** et non l'adressage historique CHS **BIOS** (cylinder-head-sector).
- (UEFI) MBR protecteur (protective MBR) : une structure qui protège les disques GPT des écritures provenant d'utilitaires disques qui ne (re)connaissent pas les informations de GPT
- (Ce n'est pas la même chose que MBR dans le BIOS)
- MBR Protecteur ->protection pour la table GPT (UEFI fonctionne avec la table GPT)
- MBR c'est table de partitionnement de BIOS

6- EFI System Partition (ESP) :

ESP (elle est dans le GPT) c'est une partition contient les fichiers de démarrage du système, les pilotes de périphériques et d'autres fichiers nécessaires pour démarrer le système d'exploitation :

- BOOTX64.EFI : Le fichier de Boot Manager (gestionnaire de démarrage efibootmgr)
- fbx64.efi : résoudre le problème des entrées NVRAM manquantes
- shimx64.efi est conçu pour démarrer GRUB2 si le secure boot (démarrage sécurisé) est activé
- mmx64.efi MOK manager(Machine Owner Key) sécurise le processus de démarrage
- grubx64.efi est le chargeur de démarrage. (Le bootloader) GRUB 2
- BOOTX64.CSV : le fichier lu lors de l'installation pour ajouter des nouvelles entrées.

7- Les étapes de séquence démarrage UEFI :

UEFI --> AutoTest (POST) -->Activer les composants basiques --> table de partitionnement GPT --> Déterminer le ESP --> Démarrage sécurisé activé ou non --> Initramfs (apartir de Vmlinuz) --> Kernel

--> system d

AutoTest (POST)	l'UEFI effectue un autotest du système (POST) pour identifier les défaillances matérielles élémentaires dès que la machine est mise sous tension.
Activer les composants basiques	L'UEFI active les composants basiques pour charger le système, comme la sortie vidéo, le clavier et les médias de stockage.
Table de partitionnement GPT	UEFI recherche une partition GPT avec un GUID spécifique qui l'identifie comme la partition système EFI (ESP)
Déterminer le ESP qui contient les fichiers de démarrages	En cas de présence de plusieurs périphériques de démarrage, le gestionnaire de démarrage UEFI détermine l'ESP approprié à utiliser, en fonction de l'ordre défini dans le gestionnaire de démarrage (boot manager)

Si le démarrage sécurisé n'est pas activé, le gestionnaire de démarrage **BOOTX64.Efi** exécute le chargeur de démarrage **grubx64.efi.**	Si le démarrage sécurisé est activé l'application **shimx64.efi** responsable de la certification est chargée en premier avant le chargeur de démarrage GRUB 2 Si le certificat est valide, le chargeur de démarrage s'exécute et, à son tour, valide le noyau qu'il est configuré pour le charger. Après il va exécuter le **grubx64.efi.**
Vmlinuz est un fichier image Extraction de contenu du Initramfs	Le chargeur de démarrage GRUB2 charge le **/boot/vmlinuz** fichier image du noyau en mémoire et extrait le contenu du **initramfs** fichier image dans un système de fichiers temporaire basé sur la mémoire (tmpfs).
Kernel (noyau)	Le noyau charge les modules nécessaires à partir de l'initramfs pour accéder au système de fichiers racine.
System d	Le noyau démarre le premier processus

8-GRUB 2 :

GRUB 2 (Grand Unified Bootloader version 2) est un chargeur de démarrage open source et populaire utilisé sur les systèmes Linux.

GRUB 2 est hautement configurable, et permet aux utilisateurs de personnaliser l'apparence de l'interface de démarrage et de choisir quel système d'exploitation ou quel noyau Linux doit être démarré par défaut.

GRUB2 est composé principalement de trois fichiers :

/etc/default/grub	un fichier contenant les paramètres du menu de GRUB 2
/etc/grub.d/	un répertoire contenant les scripts de création du menu GRUB 2, permettant de personnaliser le menu de démarrage
/boot/grub/grub.cfg	un fichier de configuration final de GRUB 2, non modifiable qui est généré automatiquement par le programme grub2-mkconfig à partir des scripts /etc/default/grub et /etc/grub.d/

/etc/default/grub

● Le fichier /etc/default/grub contient les paramètres du menu GRUB 2 qui sont lus et utilisés par les scripts sous /etc/grub.d/, ce qui permet de générer le fichier de configuration final grub.cfg.

```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical lo
cale=en_US"
```

●A connaitre :

GRUB_DEFAULT Définit l'entrée du menu de démarrage qui est l'entrée 0 sera utilisée, ce qui correspond à la première option de démarrage.

Grub_default=0 c'est la première entrée de démarrage

Grub_default=1 c'est la deuxième entrée de démarrage

Grub_default=3 c'est la 4eme entrée de démarrage

● **GRUB_TIMEOUT** est la durée en secondes de l'affichage du menu avant de lancer le système sélectionné par défaut.

● **GRUB_DISTRIBUTOR** définit la syntaxe des titres du menu (Affiche la distribution utilisé exemple Ubnutu / Debian etc... . lsb_realease

● **GRUB_CMDLINE_LINUX_DEFAULT** affichera simplement un écran vide avant d'afficher

le menu GRUB.

● **GRUB_CMDLINE_LINUX** définie les options de noyau communes. Par exemple :

✓ mem : Limite la quantité de la mémoire

✓ quiet : Masque la plupart des messages de démarrage.

✓ root : Définit la partition racine

✓ Maxcpus: Limite le nombre de processeurs (ou coeurs de processeur)

/etc/grub.d/

```
esprit@esprit-virtual-machine:~$ sudo ls -l /etc/grub.d
total 140
-rwxr-xr-x 1 root root 10627 Apr 15  2022 00_header
-rwxr-xr-x 1 root root  6260 Apr 15  2022 05_debian_theme
-rwxr-xr-x 1 root root 18683 Apr 15  2022 10_linux
-rwxr-xr-x 1 root root 43031 Apr 15  2022 10_linux_zfs
-rwxr-xr-x 1 root root 14180 Apr 15  2022 20_linux_xen
-rwxr-xr-x 1 root root  2924 Feb  6  2022 20_memtest86+
-rwxr-xr-x 1 root root 13369 Apr 15  2022 30_os-prober
-rwxr-xr-x 1 root root  1372 Apr 15  2022 30_uefi-firmware
-rwxr-xr-x 1 root root   700 Feb 19  2022 35_fwupd
-rwxr-xr-x 1 root root   214 Apr 15  2022 40_custom
-rwxr-xr-x 1 root root   215 Apr 15  2022 41_custom
-rw-r--r-- 1 root root   483 Apr 15  2022 README
```

Le dossier /etc/grub.d contient tous les scripts qui seront utilisés pour créer le fichier de configuration final grub.cfg.

● 00_header : script gérant les paramètres définis dans /etc/default/grub ;

● 05_debian_theme : script pour gérer le thème (fonds d'écran et couleurs) ;

● 10_linux : contient le script de lancement du système d'exploitation ;

● 20_memtest86+ : script permettant de générer les entrées memtest ; (semble absent de certaines installations EFI) ;

● 20_linux_xen : un script utilisé par le Bootloader pour détecter les systèmes d'exploitation Xen Linux et Xen Hypervisor µ sur le disque dur.

● 30_os-prober : contient le script de recherche des autres systèmes installés ;

● 30_uefi-firmware : script pour localiser les paramètres uefi de l'ordinateur ;

● 40_custom : configuration personnelle (systèmes à lancer en configuration manuelle, paramètres qui n'existent pas pour /etc/default/grub,..);

● 41_custom : identique à 40_custom (si on le renomme par exemple 07_custom);

9-kernel (noyau) :

Lorsque le chargeur de démarrage passe le contrôle au noyau du système, le noyau se charge de plusieurs tâches pour initialiser le système avant de lancer le premier processus. Les tâches incluent notamment :

- Réserver la mémoire
- Prendre en compte la zone d'échange (swap)
- Détecter le matériel et charger les pilotes des périphériques

- Monter le système de fichiers
- Lancer le premier processus

Partie 2 : Processus d'initialisation

Définition : Le processus d'initialisation, souvent appelé init, est le premier processus qui est lancé lors du démarrage du système d'exploitation.

Son rôle est de démarrer tous les autres processus nécessaires pour faire fonctionner le système, tels que les services système, les démons et les applications utilisateur.

Le processus d'initialisation est généralement lancé par le noyau du système d'exploitation et peut être configuré pour effectuer certaines actions spécifiques, telles que la configuration du réseau, la vérification de l'intégrité du système de fichiers ou l'exécution de scripts d'initialisation.

1-Systemd :

• Systemd (pour « system daemon » : le démon du système) est le système d'initialisation et de gestion des services adopté par les dernières distributions Linux.

• C'est un ensemble de programmes destinés à la gestion système.

Systemd fournit de nombreuses fonctionnalités :

- Le lancement en parallèle des services systèmes au démarrage du système d'exploitation,
- L'activation à la demande de démons,
- La gestion des dépendances entre les services,

2- Structure de l'unités Systemd :

La plupart des actions de systemd est effectuée grâce à des unités (units) qui sont des ressources que systemd sait gérer. Les unités sont caractérisées par un nom et un type. Les unités les plus importantes : Unité de service (service) / Unité cible (target) / Unit de montage(mout)

Type d'unité	Extension de fichier	Description
Unité du service	.service	Service système.
Unité cible	.target	Un groupe d'unités systemd.
Unité Automount	.automount	Un point Automount du système de fichiers.
Unité du périphérique	.device	Fichier du périphérique reconnu par le noyau.
Unité de montage	.mount	Point de montage du système de fichiers.
Unité de chemin	.path	Un fichier ou répertoire dans un système de fichiers.
Unité scope	.scope	Un processus créé de manière externe.
Unité de tranche	.slice	Un groupe d'unités organisées de manière hiérarchique qui gèrent des processus système.
Unité d'instantané	.snapshot	Un état enregistré du gestionnaire systemd.
Unité de socket	.socket	Un socket de communication inter-processus.
Unité swap	.swap	Un périphérique ou fichier swap.
Unité minuteur	.timer	Un minuteur systemd.

Explication:
Dans systemd, une unité est une ressource qui représente une tâche ou un processus à gérer. Les unités ont un nom et un type, et peuvent être de différents types, tels que des services, des cibles, des points de montage, des sockets, des appareils, etc.

Chaque unité a un fichier de configuration qui contient des informations sur la façon dont elle doit être gérée par systemd, comme sa dépendance à d'autres unités, ses paramètres de démarrage, son environnement, etc.

La structure de l'unité :

[UNIT]
Description :
Documentation : - Cette déclaration indique l'emplacement de la page de manuel (document d'aide).
After: - Cette mention énumère les unités qui doivent être activées avant l'unité en question.
Before: - Cette déclaration énumère les unités qui doivent être activées après l'unité en question.
Wants: - Cette déclaration liste les unités/services qui doivent être démarrés avant l'unité en question, le démarrage de ces unités/services n'est pas obligatoire pour que l'unité en question fonctionne correctement.
Requires : - Cette déclaration énumère les unités/services qui doivent être démarrés avant l'unite en question, le démarrage de ces unités/services est obligatoire .
Conflicts: - Cette déclaration énumère les unités/services qui doivent être arrêtés avant le démarrage de l'unité en question.

[TYPE] exemples : [Service] / [Socket] / [Target] / [Mount] (type de l'unité)
Type=type de service (**simple**, **forking**, **oneshot**, **dbus**, **notify**, **idle**) (type de type de l'unité)
ExecStart =commande ou chemin vers l'exécutable à lancer au démarrage
ExecReload = commande ou chemin vers l'exécutable à lancer pour recharger la configuration
Restart = détermine comment redémarrer le service en cas d'échec

[INSTALL]
WantedBy : - Nom de l'unité cible qui démarre automatiquement cette unité..

[Unit]
Description=
After = avant le démarrage
Before= après le démarrage
Wants = démarre avant mais pas obligatoire
Requires = démarré avant mais obligatoire
Conflits = doit etre arreter avant le démarrage

```
[Service]
ExecStart=/bin/bash        /usr/sbin/info_sys.sh
ExecReload=recharger config
Restart= redémarrer en cas d'échec

[Install]
Wantedby= démarrer automatiquement
```

Exemple à faire dans le TD_Systemd.

Le fichier test.service

```
[Unit]
Description=cette unite est un service test
After=network.target
Requires=display-manager.service


[Service]
ExectStart=/bin/bash /usr/sbin/hello.sh

[Install]
Wantedby=multi-user.target
```

3- Targets Systemd :

Les unités de cible se terminent par l'extension .target et leur unique but consiste à regrouper d'autres unités Systemd dans une chaîne de dépendances,

Par exemple, l'unité graphical.target, qui est utilisée pour lancer une session graphique, lance des services systèmes comme le gestionnaire d'affichage GNOME (display-manager.service) et active également l'unité multi-user.target.

Tâches / Système	Systemd
Arrêt	poweroff.target
Mono utilisateur	rescue.target
Multi-utilisateur	multi-user.target
Multi-utilisateur avec réseau	multi-user.target
Personnalisable	runlevel4.target
Multi-utilisateur avec réseau et graphique	graphical.target
Reboot	reboot.target
Changer de runlevel (Ex 3)	systemctl isolate multi-user.target
Changer le runlevel par défaut (Ex 3)	systemctl set-default multi-user.target
Vérifier le runlevel actif	systemctl get-default

4- Utilitaires de base du systemd :

Les unités de base de systemd sont les unités qui sont fournies avec le système d'exploitation et qui sont utilisées pour le démarrage, l'arrêt et la gestion des services.

Les unités de base de systemd sont stockées dans des emplacements spécifiques du système de fichiers : `/usr/lib/systemd/system` pour les unités fournies par le système et `/etc/systemd/system` pour les unités configurées par l'utilisateur.

La cible par défaut du système est la cible vers laquelle le système doit démarrer par défaut. Elle est définie par le fichier `/etc/systemd/system/default.target`

Tâches / Système	Systemd
Démarrer un service	systemctl start le service.service
Stopper un service	systemctl stop le service.service
Redémarrer un service	systemctl restart le service.service
Recharger un service	systemctl reload le service.service
Statut d'un service	systemctl status le service.service
Activer un service au démarrage	systemctl enable le service.service
Désactiver un service au démarrage	systemctl disable le service.service
vérifier si un service est activé au boot	systemctl is-enabled le service.service

Tâches / Système	Systemd
Arrêter le système	systemctl halt
Eteindre le système	systemctl poweroff
Redémarrer le système	systemctl reboot
Suspendre le système	systemctl suspend
Hiberner le système	systemctl hibernate
Afficher les logs systèmes	journalctl -f
Basculer temporairement de runlevel	systemctl isolate nom.target
Changer de runlevel par défaut	systemctl set-default nom.target

Hiberner le système signifie sauvegarder l'état actuel du système sur le disque dur et éteindre l'ordinateur.

5- Caractéristique du Systemd :

- Conception propre, moderne et efficace.
- Traitement simultané et parallèle au démarrage.
- Syntaxe d'unité simple.
- Empreintes mémoires réduites.
- Technique améliorée pour exprimer les dépendances.

6-Correction TD Systemd :

Le fichier test.service

```
[Unit]
Description = Hello exemple
After=network.target
Requires=display-manager.service

[Service]
ExecStart=/bin/bash /usr/sbin/hello.sh

[Install]
WantedBy=multi-user.target
```

Remarque : **`/usr/sbin/`** est un répertoire dans les systèmes Linux et Unix qui contient des commandes exécutables (ou des scripts) pour les administrateurs système.

```
#! /bin/bash

echo "Hello everybody, we will give you some information about your OS:" > /tmp/test.txt

n=`find /etc/grub.d -type f -perm 755 | wc -l`
echo "There are $n scripts template used with GRUB2." >> /tmp/test.txt

m=`grep ^menuentry /boot/grub/grub.cfg | wc -l`
echo "There are $m GRUB2 menu entries." >> /tmp/test.txt

k=`cut -d ':' -f 3 /etc/passwd | grep [1-9][0-9][0-9][0-9] | wc -l`
echo "There are $k simple users with UID >= 1000" >> /tmp/test.txt
```

Pour inclure le nouveau service, recharger tous les fichiers services :

Sudo systemctl realod test.service **recharger 1 seul service**

Sudo systemctl daemon-reload **recharger tout les services**

Activer maintenant le service “test” :

Sudo systemctl enable /usr/lib/systemd/system/test.service

Redémarrer la machine : reboot

Afficher le contenu du fichier : cat /tmp/test.txt

Vérifier l’état du service test :

Sudo systemctl status test.service